Transforming Transaction Models into ArchiMate

Sybren de Kinderen[1], Khaled Gaaloul[1], and H.A. (Erik) Proper[1,2]

[1] CRP Henri Tudor
L-1855 Luxembourg-Kirchberg, Luxembourg
sybren.dekinderen, khaled.gaaloul, erik.proper@tudor.lu
[2] ICIS, Radboud University Nijmegen
P. O. BOX 9010 6500, GL Nijmegen, The Netherlands

Abstract. ArchiMate, a language for modelling an organisation from a holistic perspective, lacks guidelines and techniques for exploring each of its perspectives in depth. To partly address this issue, we propose to use the DEMO modelling technique and toolset as a front-end for ArchiMate. In particular, DEMO adds to ArchiMate a conceptual clarity, as well as tools and techniques for modelling business processes.
Specifically, in this paper we contribute a formal model transformation from DEMO to ArchiMate, and show how this model transformation can be used to transform DEMO models into ArchiMate models. In addition, we provide a software implementation of this model transformation, based on the ATL model transformation environment.
Our model transformation approach is illustrated by a fictitious but realistic case study from the insurance domain.
Keywords: ArchiMate, DEMO, meta model, model transformation

1 Introduction

ArchiMate, a language for expressing Enterprise Architectures, allows for modelling an enterprise from a holistic perspective, showing amongst others an organisation's products, how these products are realised by business processes, and how in turn these processes are supported by information systems. This holistic perspective on an enterprise helps guide change processes [1], provides insight into cost structures [2], and more.

However, because of the holistic nature of ArchiMate, for each of the perspectives that ArchiMate models, it lacks the specific tools and techniques to explore these perspectives *in-depth*. For example, ArchiMate lacks guidelines for process modelling. Moreover, as pointed out by [3] ArchiMate lacks conceptual clarity. This lacking clarity is of course logical given the coarse-grained, holistic nature of ArchiMate. Nevertheless, as a result, different modellers create different models. To address the above issues, it has already been suggested that ArchiMate could benefit from the integration of other languages to explore each of its perspectives in depth [2, 3]. One such language is DEMO.

DEMO, short for Design and Engineering Methodology for Organizations, is a comprehensive set of conceptual modelling techniques focused on modelling

the *ontological* aspects of an organisation [4, 5]. With ontological, we mean that DEMO models the essential, implementation-independent aspects of an organisation only: it abstracts away from implementation-specific details, such as the information systems present in a business collaboration. Also, DEMO has proven itself practically by, amongst others, (1) aiding in the standardisation of message exchange formats in construction sector [6], (2) acting, in a governmental body, as a point of departure for business process modelling [7], and, by (3) fostering, in the aerospace industry, a shared understanding of fragmented strategic concerns, and a link of these strategic concerns to design principles [8].

In our work, DEMO forms a forms interesting complement to ArchiMate because (1) DEMO provides a clear-cut, unambiguous definition of its grammar. If used as a front-end to ArchiMate, this largely disambiguates ArchiMate's artifacts [3], and (2) DEMO provides transaction patterns, process-based patterns that aid in eliciting the main steps carried out in the business processes of an organisation.

The contribution of this paper is two-fold: (1) a formal mapping of the DEMO and ArchiMate meta models underlying these techniques, and (2) a systematic application of these meta models to map a model created in DEMO to a model of an enterprise architecture in ArchiMate. In doing so, we use a running example of an insurance scenario to illustrate our ideas. Moreover, we discuss a software tool, based upon our proposed meta models, that interfaces between two existing software environments for modelling with, respectively, DEMO and ArchiMate. This interfacing software tool provides for partial computational validation of the proposed model integration.

As this paper builds upon work wherein DEMO is used as a bridge between e^3value and ArchiMate [9], we focus only on the ontological layer of DEMO, and not its datalogical and infological layers (roughly stated, the latter two layers focus on message exchanges). Fig. 1 depicts this focus. Here, we see that we consider DEMO as a project level language that supplements ArchiMate as an enterprise architecture language, meaning that DEMO focuses on modelling one specific aspect of an organisation, supplementing the holistic ArchiMate language. Moreover, referring to Fig. 1, we see that we focus on DEMO to support ArchiMate on the ontological level, independent from implementation-specific concerns such as the supporting IT infrastructure.

Yet, relating the infological and datalogical perspectives of DEMO to ArchiMate is interesting as well, in particular because the data- and infological layers aid in modelling ArchiMate's IT infrastructure. However, mapping the info- and datalogical DEMO perspectives to ArchiMate means that we also have to overcome the difference between an implementation-specific perspective (the ArchiMate IT infrastructure layer) and ontological perspective (DEMO's info- and datalogical layers) on the organization at hand. Therefore, we opt to focus on mapping DEMO's ontologial perspective only, and to later on expand this with DEMO's datalogical and infological perspectives.

The remainder of this paper is structured as follows. Sect. 2 briefly discusses the transformation between DEMO and ArchiMate. Subsequently, this

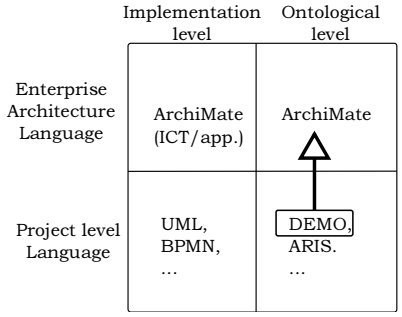

Fig. 1. This paper: DEMO supports ArchMate from an ontological perspective only, not ArchiMate's implementation specific ICT-perspective

transformation is discussed in further detail. First, illustrated by the insurance case, we show exactly how we use DEMO as a front-end to ArchiMate (in Sect. 3). Therafter, we show how to transform a DEMO model into an ArchiMate model(Sect. 4), and discuss our implementation of this transformation in a software tool (Sect. 5). Sect. 6 presents related work. Finally, Sect. 7 concludes.

2 Transforming DEMO transaction models into ArchiMate

In this section, we show the model transformation of DEMO into ArchiMate (see Fig. 2). In line with [3], we use DEMO as a front-end for ArchiMate. First, based on the DEMO meta model, we explore an organisation, starting from the high-level transactions that it performs, and gradually refining these into detailed business processes that realise these processes. So, conform to Fig. 2, we create a DEMO instantiation for a DEMO meta model. Thereafter, from a ruleset for mapping the DEMO and ArchiMate meta models (the transformation definition) and a specific application of these rules to the DEMO meta model (specified by the transformation engine), we derive an ArchiMate meta model, and a meta model instantiation.

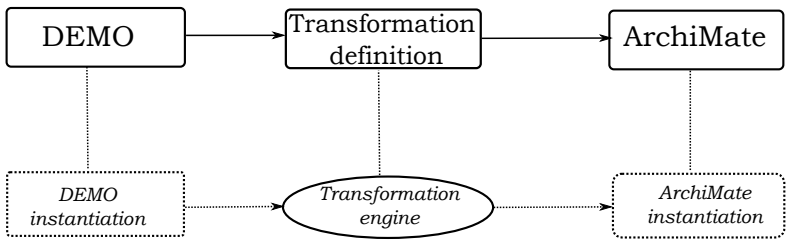

Fig. 2. Transforming DEMO transaction models into ArchiMate, adapted from [10]

3 Modelling insurance transactions and processes in DEMO

In this section, we create a model according to the DEMO meta model and its accompanying tools, so that we have a solid basis for creating an Enterprise Architecture model in ArchiMate (which we discuss in Sect. 4). First, in Sect 3.1, we introduce the running insurance case that will be used for illustrating our application of DEMO and, later on, the transformation into ArchiMate. Thereafter, we apply DEMO to our running case. First, we create a high-level DEMO model that focuses on organisational transactions (in Sect. 3.2). Following this, we detail the business processes realising these organisational transactions (in Sect. 3.3).

3.1 Archinsurance: selling car insurance via insurance brokers

For illustration purposes, we use the insurance company Archinsurance as a fictitious but realistic use case. This case is grounded in the insurance case that is used to illustrate the ArchiMate modelling language [1, 11], as well as a paper on the function of insurance intermediaries [12].

For this paper we focus on car insurance, an insurance product that Archinsurance sells via *insurance brokers*. The main reason for selling insurance via brokers is to reduce the risk of *adverse risk profiles* [12], incomplete or faulty risk profiles of customers that lead insurance companies to sell inappropriate insurance packages. To mitigate adverse risk profiles, insurance companies may therefore rely on insurance brokers, whose core business it is to match customer profiles to appropriate packages.

3.2 The DEMO' meta model and transaction model

We use DEMO to model the sale of car insurance by Archinsurance. As stated, DEMO aims at modelling the ontological, implementation-independent aspects of an organisation only. DEMO achieves its focus on ontological aspects by perceiving of an organisation as a *social* system of actors, that collaborate to achieve a common goal. Chief to this collaboration are acts: production acts, and communication acts. Production acts bring about (part of) a good or service, and directly contribute to achieving the organisation's common goal. In the Archinsurance case, a production act is for example 'Find matching insurance package', as executed by the insurance broker on behalf of the customer. Communication acts, then, serve to coordinate among the actors that either receive results from, or execute, the production acts. In the Archinsurance case, 'Apply for insurance' is for example a communicative act used by the customer to indicate to the insurance broker the interest in an insurance package.

In this paper, we use only a subset of the DEMO conceptualisation and techniques, referred to as DEMO', or DEMO derived. DEMO' borrows from DEMO a subset of concepts, to which we refer as the DEMO' meta model, and the DEMO standard transaction pattern.

The DEMO' meta model is depicted in figure 5. For the Archinsurance case, Fig. 3 presents an instantiation of this meta model, referred to as a DEMO' transaction model. Here, we see the high-level transactions that together implement the selling of car insurance via an insurance broker. First, the transaction 'create customized insurance package', whereby the broker matches a customer profile to a fitting insurance package and, second, the transaction 'contracting', whereby the car insurance department within Archinsurance - based on the risk profile that it receives from the broker - underwrites the insurance. This (underwriting) means that the car insururance department creates an insurance package for the customer, and calculates the associated premium. These transactions are carried out by physical subjects (for example, the 'Car insurance department') that carry out an organisational role (for example: the 'Car insurance department' fulfills the role 'underwriter').

Note however, that a DEMO' transaction model does not show in detail the business processes that realise the modelled transactions.

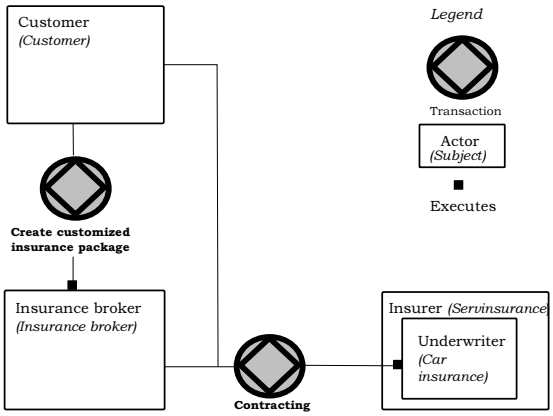

Fig. 3. DEMO transaction model of Archinsurance

3.3 From an Archinsurance transaction model to a process model

To elucidate the business processes underlying the DEMO' transactions, DEMO' borrows from DEMO transaction patterns. The DEMO' standard transaction pattern focuses on a process-based pattern of (instantiations of) DEMO' meta model concepts, showing the sequence of acts that *always* needs to be executed to realise an economic transaction. So, here we see again DEMO's emphasis on the ontological aspect of an organisation: no matter what the domain, if we perceive of an organisation as a social entity, then we see a pattern of generic acts

that *always* occurs in carrying out a transaction [5]. So, for example, one actor always has to initiate a transaction by performing the act 'request' (which in the Archinsurance case may translate to the act 'Apply for insurance' as carried out by a customer), while another actor has to always perform the 'execute' act in order to produce the good or service that the initiating actor is interested in (in the Archinsurance case, this may translate to the act 'Find matching package' which, as mentioned before, is executed by the insurance broker).

Fig. 4 shows an instantiation of the DEMO' standard transaction pattern for the Archinsurance case. Note here in particular that the illocutionary acts from the transaction patterns, such as request, promise and accept, aid in detailing exactly what business process steps- or: DEMO' acts - together realise the transactions in Archinsurance's transaction model (see Fig. 3). For example, for the Archinsuranc case, the act 'request' translates to the act 'Apply for insurance', an act carried out by the customer to trigger the insurance process, while the act 'execute' translates to the act 'find matching package' as carried out by the insurance broker.

4 Translating DEMO' process models to ArchiMate

In this section, we introduce the ArchiMate modelling language and focus on its business layer meta model. Thereafter, we present the mapping between the DEMO' and the ArchiMate business layer meta models. Subsequently, we apply this mapping to transform the DEMO' process model of Archinsurance to an ArchiMate model.

4.1 The ArchiMate business layer meta model

We rely on the ArchiMate modelling language to model the enterprise architecture of the Archinsurance case. ArchiMate has been transferred to the Open Group, where it is slated to become the standard for architectural description accompanying the Open Group's architecture framework TOGAF [13]. As stated, ArchiMate offers a coherent, holistic, description of the enterprise architecture to enable communication among stakeholders, and to guide change processes within Archinsurance.

We identify the main concepts for architectural descriptions that can be placed in the business layer of the ArchiMate meta model. Fig. 5 gives an excerpt (ArchiMate') of the business layer concepts and their relationships. The business layer refers to the static structure of an organisation, in terms of the entities that make up the organisation and their relationships [1].

4.2 Mapping the DEMO' meta model to the ArchiMate' meta model

For mapping DEMO' to ArchiMate, we use the meta model mapping technique described in [14] where authors distinguished different types of mappings, the

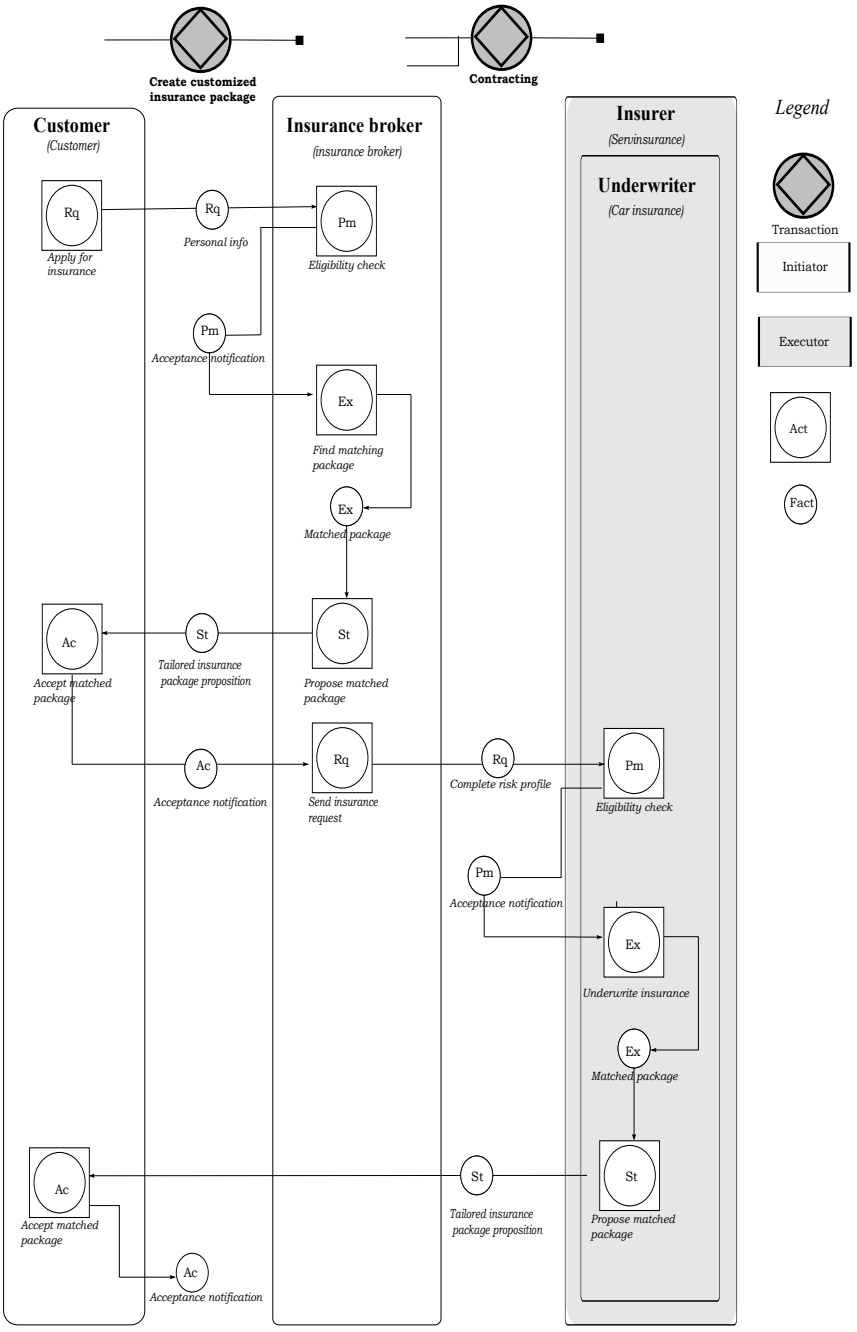

Fig. 4. DEMO Business process model, detailing the Archinsurance transaction model

most relevant for our work being (1) *class-to-class mappings*, which relates a concept from meta model A to a concept from meta model B (e.g., a 'Subject' from DEMO' relates to an 'Actor' from ArchiMate'). And (2) *relation-to-relation mappings*, which relates concept relationships from meta model A with concept relationships from meta model B (e.g., 'performs_role' between the concepts Subject and Actor from DEMO' relates to the ArchiMate relation 'assigned_to' between the concepts Actor and Business role).

[14] also distinguishes between different types of relations, the most important for us being: equivalence, generalisation of, and its inverse is specialisation of, and no relation.

Now that we have explained the main ideas behind ArchiMate and presented an excerpt of the business layer meta model, we translate a DEMO process model for Archinsurance into an ArchiMate business layer model. We do this in two main steps: (1) Translate the concepts from a DEMO process model to an ArchiMate process model, which we can do given that, looking at the concept definitions, the holistic ArchiMate language subsumes DEMO's social perspective, (2) Define a (partial) enterprise architecture model from a business perspective that focuses on the DEMO' process model. Here, we construct an ArchiMate model from the mapped DEMO' concepts. As we now actually construct an ArchiMate model, we take here into consideration (a) the difference in abstraction level between DEMO and ArchiMate, and (b) additional ArchiMate constructs not present in DEMO', for example for depicting an IT perspective on the organisation at hand.

Step 1: Horizontal integration via meta model mapping The first step will apply our mapping between the DEMO' meta model and the ArchiMate' business layer meta model. Here, we make a mapping on a purely *horizontal* level (cf. [14]), meaning that we consider only differences between aspects modelled in DEMO and ArchiMate *on the same abstraction level*. In doing so, we apply the DEMO' - ArchiMate' meta model mapping from Fig. 5, and the corresponding rationale of our meta model mapping (i.e., Table 4.2). In Fig. 5, we define a specialisation relation between the mapped concepts from DEMO' to ArchiMate' concepts. Here, we assume - based on the concept definitions from DEMO and ArchiMate - that the holistic ArchiMate language encompasses the specific social perspective emphasised by DEMO'.

For Archinsurance, we apply this mapping as follows. For reference, see the Archinsurance ArchiMate model in Fig. 6, and the Archinsurance DEMO' process model in Fig. 4.

- *Subjects from DEMO' map to business actors in ArchiMate'.* We define a mapping relation from DEMO' to ArchiMate' concepts where a subject performing a role in DEMO' is an actor in ArchiMate'. For instance, we map the subject 'car insurance' (a department within Archinsurance specialised in car insurances) to the business actor 'car insurance' in ArchiMate'.
- *Actors in DEMO' map to business role in ArchiMate'.* An actor performing an act in DEMO' is the associated role to a business actor in ArchiMate'. For

instance, we map the actor 'Underwriter', an associated role to the subject 'car insurance', to the role 'Underwriter' in ArchiMate'.
- *An act from DEMO' is mapped to a set of business behaviour/events in Archimate'.* An act performed by an actor in DEMO' maps to a business process step in ArchiMate'. For instance, we map the act 'Find matching package' to the same business step in ArchiMate'.
- *Transactions in DEMO' map to business interactions in Archimate'.* A transaction is a collection of acts in DEMO' and a business interaction includes a set of business steps performed within a collaboration in ArchiMate'. For instance, we map the transaction 'Create customized insurance package' to the business interaction in ArchiMate'.

In addition, we perform relation-to-relation mapping between DEMO' and ArchiMate (see Table 5). As such, we relate:

- *The relation (Subject)performs_role(Actor) from DEMO' to the relation (Business actor)assigned_to(Business role) from ArchiMate.* For example, in both DEMO' and ArchiMate, the department 'Car insurance' performs the role of 'Underwriter'.
- *The relation (Transaction)consists_of(Act) from DEMO' to the relation (Business collaboration)triggers(Business event/business behaviour) from ArchiMate.* For instance, both in DEMO and ArchiMate, 'Create customised insurance package' consists of the more elementary acts 'apply for insurance' and 'find matching package'.
- *The relation (Actor)performs(Act) from DEMO' to the relation (Role)assigned_to (Business event/business behaviour) from ArchiMate.* For example, in both DEMO' and ArchiMate, the underwriter carries out the act 'Underwrite insurance'.

Step 2: Vertical integration: defining an appropriate abstraction level in ArchiMate The second step consists of defining an enterprise architecture model using ArchiMate to represent a DEMO' process model.

First, in addition to the horizontal differences in Step 1, we now consider also the *vertical* differences between DEMO' and ArchiMate'. This means that we remove from the ArchiMate model any elements that are too detailed for depicting a holistic perspective on the organisation at hand. For example: for the the Archinsurance case, we thus remove the business objects 'acceptance notification' (which ArchiMate inherits from the DEMO process model in Fig. 4), since they are too detailed for the high-level model overview provided by ArchiMate.

Second, we supplement the model elements inherited from DEMO' with ArchiMate constructs. This we do to fully express a holistic perspective on the organisation at hand, most prominently in terms of the supporting IT infrastructure. For example, as we can see in Fig. 6, for Archinsurance we model that the business process activities 'eligibility check' and 'underwrite insurance' are

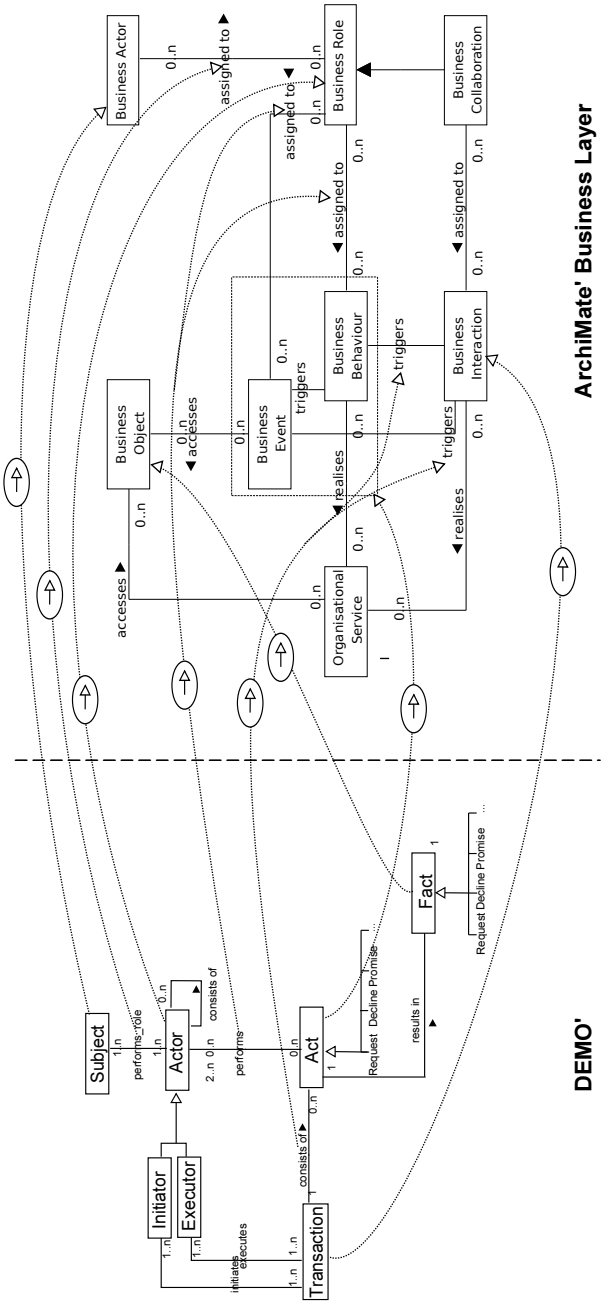

Fig. 5. Mapping of DEMO' and ArchiMate' meta models

DEMO' concepts	ArchiMate' concepts	Mapping rationale
Actor	Business role	In DEMO, an actor refers to a social role played by a subject in an organisation. Such a social role corresponds to the definition of a business role in ArchiMate where roles are typically used to distinguish responsibilities.
Subject	Business actor	A DEMO subject is an organisational entity - person, department or otherwise - that can fulfil an organisational role. This corresponds to a business actor in ArchiMate, which is an organisational entity that performs some behaviour (cf. [1]), thus it can also fulfil a role.
Act	Business behaviour/event	An act is performed by a subject in a social role. Its scope is about contribution/coordination for services. In the ArchiMate context, it corresponds to the realisation of an organisational service via a business process or a function (business behaviour) or a business event (e.g., an external request).
Transaction	Business interaction	For DEMO transactions, the initiation and execution are performed by different actors. This emphasises the interaction aspect that we can find in ArchiMate, where a business interaction is carried out by more than one actor.
Fact	Business object	A fact is any object that results from performing an act. In ArchiMate', this corresponds to a business object, which 'represent the important concepts in which the business thinks about a domain' [1].

Table 1. DEMO' - ArchiMate' meta model concept mapping relations

DEMO' relation	ArchiMate' relation	Mapping rationale
performs_role	assigned_to	In both DEMO and ArchiMate, one relates a real world entity (e.g., Archinsurance) to a role played by that entity (e.g., the role of insurer in the case of Archinsurance).
consists_of	triggers	As transactions map to business interactions, and acts map to business events and business behaviour, the relation 'consists_of' between transactions and acts in DEMO maps logically to the relation 'triggers' between business interactions and business events/business behaviour in ArchiMate.
performs	assigned_to	While both use different nomenclature, in both DEMO and ArchiMate, a role - not the real-world entity behind it - carries out acts.

Table 2. DEMO' - ArchiMate' meta model relations mapping

supported by a risk assessment application, and that both the business collaboration 'create customized insurance package' is supported by administrative applications from both the insurance broker and Archinsurance.

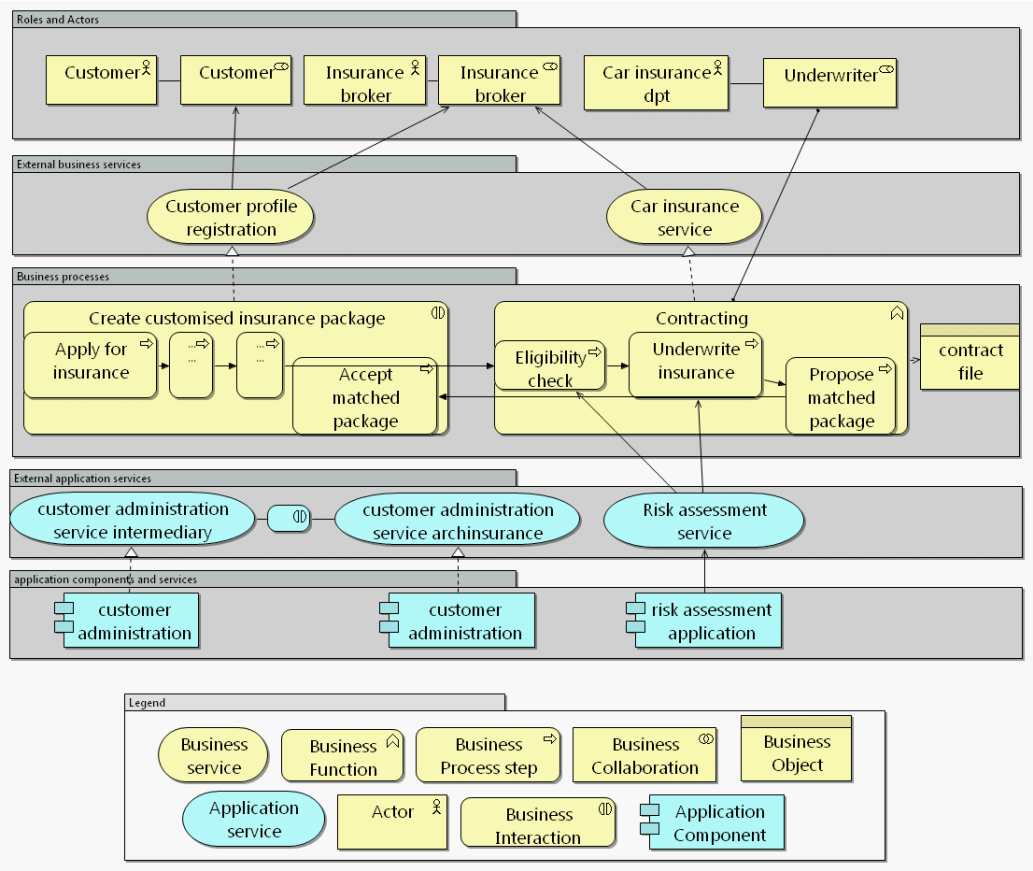

Fig. 6. (Partial) enterprise architecture model based on DEMO process

5 Tool implementation

We have implemented our DEMO to ArchiMate mapping in ATL [3], an Integrated Development Environment for implementing model transformations that is built on top of the eclipse platform. This mapping conforms exactly to the mapping defined in Fig. 5: no concepts are added, modified, or removed.

[3] http://www.eclipse.org/atl/

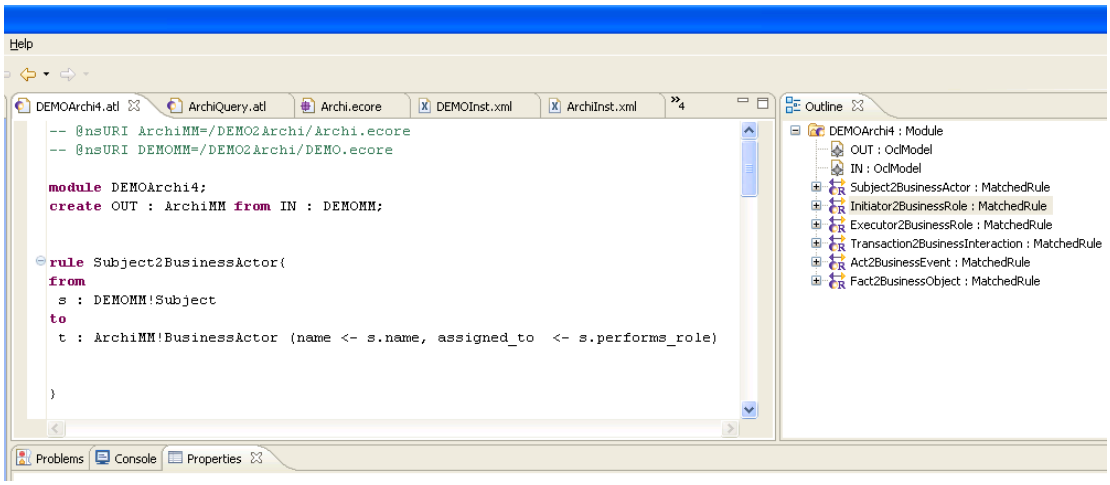

Fig. 7. Mapping DEMO to ArchiMate in the ATL model transformation environment (excerpt)

Table 5 shows a sample of model instantiations in XML, for DEMO (left) and ArchiMate (right). ATL can produce an ArchiMate instantiation in XML, given: (1) the DEMO and ArchiMate meta models, defined in an ECORE syntax; (2) a meta model mapping, defined in ATL (see fig. 7), and of course: (3) an instationation of the DEMO meta model, defined in XML.

DEMO' instantation	ArchiMate instantation
<BusinessActor name="Insurance Broker"> <assigned_to name="Insurance Broker Role"/> </BusinessActor> <BusinessActor name="Customer"> <assigned_to name="Customer Role"/>	<Subject name="Insurance Broker"> <performs_role name="Insurance Broker Role"/> </Subject> <Subject name="Customer"> <performs_role name="Customer Role"/> </Subject>

Table 3. Sample of the XML instantiations for the Archinsurance case, in DEMO (ex-ante model transformation) and ArchiMate (ex-post model transformation)

6 Related Work

The e^3*alignment* approach provides tools for actually creating business-ICT alignment. It does so by ensuring that conceptual models depicting a strategic,

value, process and ICT perspectives respectively on the value web at hand are consistent with one another [15]. However, this approach works only on a syntactic level. For instance, if the concept of an actor in e^3value and the concept of a swim lane in an UML activity diagram means the same is not a consideration. Derzsi et al. enable profitability calculations of an ICT-infrastructure by providing a meta model that links an IT infrastructure modelled in UML to e^3value [16]. This approach has more formality than $e^3alignment$, yet it focuses on a link between IT and value only. As a result, business processes are not a consideration while these are realistically cost carriers as well.

Ontological merging approaches address the semi-automated integration of system models [17]. System models are created in terms of modelling language, which in itself is based on a meta model. Syntactic and semantic mapping between pairs of meta models has been facilitated by the application of existing approaches for ontology mapping [18, 19]. Ontologies improve not only the semantics of a meta model but also provide a potential way in which these meta models can be bridged with each other to be integrated within a common context [20]. However, ontology mapping approaches such as [18, 19] focus on providing an approximation of a mapping between two ontologies. Yet, in our research we require a precise mapping. Since our starting point are ontologies, such as DEMO with relatively few concepts (compared to larger ones such as found in the medical domain), it seems better to perform mapping/integration manually and as such, avoid an approximation of a mapping.

7 Conclusion and Future Work

In this paper, we used DEMO as a front end for ArchiMate. Using a case from the insurance domain, introduced a mapping between DEMO and ArchiMate, and showed how this mapping can be applied to translate a DEMO model into an ArchiMate model. Also, we showed why such a mapping makes sense, in particular by using the transaction patterns from DEMO for constructing a business process that is later transformed into ArchiMate. Finally, we discussed an implementation of our mapping in the model transformation language ATL.

For further research, we will also consider the data- and infological layers of DEMO, more specifically how these compare to the implementation-specific ICT-infrastructure layer of ArchiMate. Also, we will look into enriching ArchiMate itself with other techniques, by for example borrowing concepts from e^3value to add expressivity from a value perspective. This naturally introduces a number of research challenges, such as how to balance model integration - changing ArchiMate itself - with model transformation - leaving a concern to a specifc technique, and import results into ArchiMate.

References

1. H. Jonkers, M.M. Lankhorst, R. van Buuren, S.J.B.A. Hoppenbrouwers, M. Bonsangue, and L. Van der Torre. Concepts for Modeling Enterprise Architectures. *International Journal of Cooperative Information Systems*, 13(3):257–288, 2004.

2. R. van Buuren, J. Gordijn, and W. Janssen. Business case modelling for e-services. In *18 th Bled eConference eIntegration in Action*, 2005.
3. R. Ettema and J.L.G. Dietz. Archimate and demo–mates to date? *Advances in Enterprise Engineering III*, pages 172–186, 2009.
4. J.L.G. Dietz. *Enterprise ontology: theory and methodology*. Springer Verlag, 2006.
5. J.L.G. Dietz. The deep structure of business processes. *Communications of the ACM*, 49(5):58–64, 2006.
6. Stichting DEMO kenniscentrum. Demo: The visi case, http://www.demo.nl/practical-case-studies/why-visi. last accessed on 5 December.
7. M. Op 't Land, K. Middeljans, and V. Buller. Enterprise Ontology based Application Portfolio Rationalization at Rijkswaterstaat. In *The 4th Dutch Championship ICT Architecture*, 2007.
8. Stichting DEMO kenniscentrum. Demo: The klm case, http://www.demo.nl/attachments/article/21/080610_Klantcase_KLM.pdf. last accessed on 5 December.
9. S. de Kinderen, K. Gaaloul, and E. Proper. Integrating value and transacton modelling into archimate. 2011.
10. K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006.
11. M.M. Lankhorst, H.A. Proper, and H. Jonkers. The Architecture of the ArchiMate Language. *Enterprise, Business-Process and Information Systems Modeling*, pages 367–380, 2009.
12. J.D. Cummins and N.A. Doherty. The economics of insurance intermediaries. *The Journal of Risk and Insurance*, 73(3):359–396, 2006.
13. H. Jonkers, H.A. Proper, and M. Turner. TOGAF and ArchiMate: A Future Together. White Paper W192, The Open Group, November 2009.
14. S. Zivkovic, H. Kuhn, and D. Karagiannis. Facilitate modelling using method integration: An approach using mappings and integration rules. 2007.
15. V. Pijpers, J. Gordijn, and H. Akkermans. e3alignment: Exploring inter-organizational alignment in networked value constellations. *International Journal of Computer Science & Applications*, page 59, 2009.
16. Z. Derzsi, J. Gordijn, and K. Kok. Multi-perspective assessment of scalability of it-enabled networked constellations. In Ralph H. Sprague, editor, *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 492. IEEE CS, 2008.
17. V. Devedzić. Understanding ontological engineering. *Commun. ACM*, 45:136–144, April 2002.
18. M. Ehrig and S. Staab. Qom–quick ontology mapping. *The Semantic Web–ISWC 2004*, pages 683–697, 2004.
19. N.F. Noy and M.A. Musen. The prompt suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
20. H. Happel and S. Seedorf. Applications of ontologies in software engineering. In *In 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), held at the 5th International Semantic Web Conference (ISWC 2006)*, 2006.